Solution Code

```
/* C++ Program of Templated class derived from Non-templated class

#include <iostream>
#include <string>
#include<math.h>
using namespace std;

double M_PI = 3.14;

enum eColor { none = 0, red, white, blue, yellow, green, black };

class Color
{
public:
    Color(eColor color);
    void setColor(eColor color);
    eColor getColor() { return mColor; };
    std::string getStrColor();

protected:
    eColor mColor;
};

Color::Color(eColor _color)
{
   mColor = _color;
}

void Color::setColor(eColor _color)
{
    mColor = _color;
}
```

Solution Code

```
std::string Color::getStrColor()
{
    switch(mColor)
    {
        case red:
            return "red";
        case white:
            return "white";
        case blue:
            return "blue";
        case yellow:
            return "yellow";
        case green:
            return "green";
        case black:
            return "black";
        case none:
        default:
            return "none";
    }
}


template <typename T>
class Circle : public Color
{
public:
    Circle(T centerX, T centerY, T radius, eColor color);
    Circle(T centerX, T centerY, T radius);
    Circle(T radius);
    Circle();
```

Solution Code


```
  T area();
    T circumference();
    T getX();
    T getY();
    T getRadius();

protected:
    T x;
    T y;
    T radius;
};

template <typename T>
Circle<T>::Circle(T _x, T _y, T _radius, eColor _color)
: Color(_color)
{
    x = _x;
    y = _y;
    radius = _radius;
}

template <typename T>
Circle<T>::Circle(T _x, T _y, T _radius)
: Color(none)
{
    x = _x;
    y = _y;
    radius = _radius;
}
```

Solution Code


```
template <typename T>
Circle<T>::Circle(T _radius)
: Color(none)
{
    x = static_cast<T>(0);
    y = static_cast<T>(0);
    radius = _radius;
}

template <typename T>
Circle<T>::Circle()
: Color(none)
{
    x = static_cast<T>(0);
    y = static_cast<T>(0);
    radius = static_cast<T>(1);
}

template <typename T>
T Circle<T>::area()
{
    return M_PI * radius * radius;
}
```

Solution Code


```
template <typename T>
T Circle<T>::circumference()
{
    return static_cast<T>(2) * M_PI * radius;
}



int main(int argc, char* argv[])
{
    Circle<float> circleA(0.0, 0.0, 10.0, white);
    cout << "\nArea of Circle A :: "  << circleA.area() << endl;
    cout << "\nColor of Circle A :: " << circleA.getStrColor() << endl;
return 0;
}
```